LLNL-CONF-663239

LAWRENCE LIVERMORE NATIONAL LABORATORY

Accelerating Newton's method

L. L. LoDestro, J. Yao

October 23, 2014

56th Annual Mee1ng of the APS/DPP
New Orleans, LA, United States
October 27, 2014 through October 31, 2014

Accelerating Newton's method

L. L. LoDestro and J. Yao

Lawrence Livermore National Laboratory, Livermore, CA, 94551

Presented at the 56th Annual Meeting of the APS/DPP
New Orleans, LA
Oct. 27—31, 2014

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

ABSTRACT

In JCP **267**, 2014, J. Yao showed how to add a single function call to an n^{th}-order iterative algebraic solver thereby raising its order of convergence to $2n$-1. Here we generalize the scheme to arbitrarily high order, without extra derivative evaluations; and we discuss the efficiency of the schemes. For n=2 (Newton's method) and moderately large system-size M, we find a computational speed several times faster than Newton's method.

Outline

- Add a single function-evaluation to raise the order of convergence (OoC) of a root-solver from *n* to 2*n*-1.
- Efficiency and CPU time.
- Multi-step method: raise OoC to *s*(*n*-1)+1.
  - A "step" here is one new estimate of the root per iteration cycle; e.g., Newton's method is *n*=2, *s*=1.
- Results:
  - Relative efficiency cf. Newton as function of *s* and system-size *M*.
  - Example: tri-diagonal Jacobian
  - Example: Jacobian with a large condition-number
- Discussion

Add a single function-evaluation to raise the OoC from n to $2n$-1*

- Begin with an iterative algebraic solver for $f(x)=0$.
  - $f_k \equiv f(x_k)$; subscript (k) denotes iteration index.
  - For OoC n, $|x_{k+1} - x_k| = O(|x_k - x_{k-1}|^n)$.
  - Taylor series $\Longrightarrow f_k = O(|x_{k+1} - x_k|)$.
- The motivating example---raising Halley's method, n=3, to 5th order---is presented on the following page.
- The demonstration for general n is similar but with the original scheme cast in fixed-point iterative form.

*[Yao, JCP **267** (2014), 139--145]

There are two steps with this method and we demonstrate the procedure here for the case $n=3$ with the well-known Halley's method [3]. Let x_k be the kth estimate for the root. One solves the equation (assuming $f'' \neq 0$)

$$f(x_k)+f'(x_k)\delta+\frac{1}{2}f''(x_k)\delta^2=0, \tag{1}$$

and the two roots are explicitly expressed as

$$\delta=-\frac{1}{f''(x_k)}\left(f'(x_k)\pm\sqrt{f'(x_k)^2-2f(x_k)f''(x_k)}\right).$$

To recover Newton's method [2] when the quadratic term vanishes, we pick only one root and it can be written as

$$\delta=\frac{\operatorname{sgn}(f'(x_k))}{f''(x_k)}\left(\sqrt{\left(f'(x_k)\right)^2-2f(x_k)f''(x_k)}-\left|f'(x_k)\right|\right).$$

The above step uses three functional calls. Note that a Taylor series for $f(x+\delta)$ at $x=x_k$ using Eq. (1) implies

$$f(x_k+\delta)=O\left(\delta^3\right). \tag{2}$$

The next step uses one more function call to gain two more orders of convergence. One adds a term $f(x_k+\delta)$ to Eq. (1), and solves

$$f(x_k+\delta)+f(x_k)+f'(x_k)\Delta+\frac{1}{2}f''(x_k)\Delta^2=0. \tag{3}$$

The solution is similar to that obtained in the first step:

$$\Delta=\frac{\operatorname{sgn}(f'(x_k))}{f''(x_k)}\left(\sqrt{\left(f'(x_k)\right)^2-2\left(f(x_k+\delta)+f(x_k)\right)f''(x_k)}-\left|f'(x_k)\right|\right).$$

Finally, let $x_{k+1}=x_k+\Delta$ for completion of the current iteration cycle.

One computes only four function values $f(x_k)$, $f'(x_k)$, $f''(x_k)$, and $f(x_k+\delta)$. However, the above scheme is fifth order convergent as shown next.

From a Taylor expansion one obtains

$$f(x_k+\Delta)=f(x_k)+f'(x_k)\Delta+\frac{1}{2}f''(x_k)\Delta^2+\frac{1}{6}f'''(x_k)\Delta^3+O\left(\Delta^4\right).$$

The sum of the first three terms in the right-hand side is equal to $-f(x_k+\delta)$ from Eq. (3); thus $f(x_k+\Delta)=-f(x_k+\delta)+f'''(x_k)\Delta^3/6+O(\Delta^4)$. However, from Eq. (1) and the Taylor expansion of $f(x_k+\delta)$, the above estimate becomes

$$f(x_k+\Delta)=\frac{1}{6}f'''(x_k)\left(\Delta^3-\delta^3\right)+O\left(\Delta^4-\delta^4\right)=(\Delta-\delta)O\left(\Delta^2,\delta^2\right). \tag{4}$$

By subtracting Eq. (1) from Eq. (3) one arrives at

$$(\Delta-\delta)\left(f'(x_k)+O(\delta)\right)=-f(x_k+\delta)=O\left(\delta^3\right). \tag{5}$$

It tells us that Δ and δ are of the same order and

$$(\Delta-\delta)=O\left(\delta^3\right).$$

One easily sees from Eq. (4) and Eq. (5) that

$$f(x_{k+1})=f(x_k+\Delta)=O\left(\Delta^5\right).$$

Therefore the method is *fifth-order* convergent; however it employs only *four* function values. The proof above can be generalized for arbitrary n.

Efficiency of function-evaluations and CPU cost

- Higher-order methods require more *f*-evaluations $\Longrightarrow$ often *not* more computationally efficient:
  - Two iterations reduce the error by $(f_k{}^n)^n \sim f_k{}^{(n\wedge 2)}$, not $\sim f_k{}^{2n}$.
    - n^n: fairly rapid error-reduction even at low n.
    - If f is expensive it can be cheaper to take more iterations than to accomplish more per more costly iteration.
- $\mathcal{E}_f$, the error-reduction per function-evaluation, was introduced by early authors to analyze efficiency.

Efficiency and CPU cost, cont.

- $\mathcal{E} \equiv$ error-reduction exponent after k iterations: $\mathcal{E} = n^k$.
- $N_T \equiv$ total no. of f-evaluations to reach a given $\mathcal{E}$:
  $N_T = kN_1$, where N_1 is no. of f-evaluations per iteration.
- Then $\mathcal{E} = n^{N\_T/N\_1} = (n^{1/N\_1})^{N\_T}$, so that $\mathcal{E}_f = n^{1/N\_1}$.
  - $M = 1$ examples (assume cost of f', etc. ~ f):
    - Any 1-step method: Taylor series for $f \Longrightarrow \mathcal{E}_f = n^{1/n}$.
    - Page 4: $\mathcal{E}_f = (2n\text{-}1)^{1/(n+1)}$.
- C_{CPU}, $C_{\text{CPU}f} \equiv$ computational cost to realize $\mathcal{E}$, evaluate f:
  - $C_{\text{CPU}} = N_T C_{\text{CPU}f} = C_{\text{CPU}f} \ln \mathcal{E} / \ln \mathcal{E}_f \longrightarrow_{(\mathcal{E}\_f \sim 1)} C_{\text{CPU}f} \ln \mathcal{E} / (\mathcal{E}_f - 1)$.

Multi-step method: raise OoC to *s*(*n*-1)+1

III. GENERALIZATION OF THE ACCELERATION SCHEME TO FIXED-POINT ITERATIVE METHODS

Acceleration of $n > 2$ Taylor-series-based root-solvers, while achieving convergence of order $2n-1$ with $n+1$ function-evaluations, has the drawback in common with the original solver that roots of polynomials of degree $n-1$ must be solved for the intermediate and final step-sizes; and the appropriate roots from these solves must be identified. Here we circumvent these complications by developing the acceleration method for fixed-point iteration: One solves for $f(x)=0$ by iterating upon x_k according to

$$x_{k+1} \;=\; x_k + \delta_k \tag{2}$$

with $\delta_k = \delta(f_k, f'_k, f''_k, \ldots)$, where $f_k \equiv f(x_k)$, $f'_k \equiv f'(x_k)$, $f''_k \equiv f''(x_k)$, etc. Without loss of generality, δ can be written in the form

$$\delta \;=\; -\frac{f}{f'}(1+g)\,, \tag{3}$$

with $g = g(f, f', f'', \ldots)$ and $g_k \equiv g(f_k, f'_k, f''_k, \ldots)$. The iteration scheme is said to be n^{th}-order convergent if

$$f_{k+1} \;\sim\; \mathcal{O}(\delta_k^n)\,. \tag{4}$$

The Taylor expansion of f_{k+1} then becomes (employing the alternate notation $f^{(i)}$ for the i^{th} derivative of f where convenient):

$$f_{k+1} = \sum_{i=0}^{\infty} f_k^{(i)} \frac{\delta_k^i}{i!} = -f_k g_k + \sum_{i=2}^{\infty} \frac{f_k^{(i)}}{i!}\left(\frac{-f_k}{f'_k}\right)^i (1+g_k)^i\,. \tag{5}$$

In the second equality, note that $f_k g_k$, the remainder of the two lowest-order terms, can be at most of second order (since $f_{k+1} \sim \mathcal{O}(\delta_k^n)$), which then implies $f_k \sim \delta_k$, which in turn restricts g_k to at most first order. The series is in the form of a function, h, that is analyzed in the Appendix. It is shown there what conditions g must satisfy in order that $f_{k+1} \sim \mathcal{O}(\delta_k^n)$.

We now apply our acceleration approach to this fixed-point iteration. The functions f, g, and δ will remain the same. To distinguish the modified iterates, we will use use tilde's, i.e., $\tilde\delta_k \equiv \delta(\tilde f_k, \tilde f_k{}', \tilde f_k{}'' \ldots)$, $\tilde f_k \equiv f(\tilde x_k)$, $\tilde f_k{}' \equiv f'(\tilde x_k)$, $\tilde g_k \equiv g(\tilde f_k, \tilde f_k{}', \tilde f_k{}'' \ldots)$, etc. We set

$$\tilde x_{k+1} = \tilde x_k + \Delta_k$$

with

$$\Delta_k = -\frac{\tilde f_k + f^*}{\tilde f_k{}'}(1+g^*)$$
$$g^* \equiv g(\tilde f_k + f^*, \tilde f_k{}', \tilde f_k{}'' \ldots)$$

and

$$f^* \equiv f(\tilde x_k + \tilde\delta_k) = \sum_{i=0}^{\infty} \tilde f_k{}^{(i)} \frac{\tilde\delta_k^i}{i!} = -\tilde f_k \tilde g_k + \sum_{i=2}^{\infty} \frac{\tilde f_k{}^{(i)}}{i!}\left(\frac{-\tilde f_k}{\tilde f_k{}'}\right)^i (1+\tilde g_k)^i\,.$$

We then have

$$\tilde{f}_{k+1} = \sum_{i=0}^{\infty} \tilde{f}_k^{\,(i)} \frac{\Delta_k^i}{i!} = -f^* - (\tilde{f}_k + f^*)\, g^* + \sum_{i=2}^{\infty} \frac{\tilde{f}_k^{\,(i)}}{i!} \left(\frac{-(\tilde{f}_k + f^*)}{\tilde{f}_k^{\,\prime}} \right)^i (1+g^*)^i \,. \tag{6}$$

In the second equality, observe that both f^* and the remaining terms are each in the form of the function h introduced as the right-hand-side of Eq. (5). Thus the results of the Appendix can be applied to them. This gives their scalings as order $\tilde{f}_k^{\,n}$ and $(\tilde{f}_k + f^*)^n \sim \tilde{f}_k^{\,n}$ respectively, and Eq. (A7) is then used to obtain

$$\tilde{f}_{k+1} = \mathcal{O}(f^* \times \max(f^*, \tilde{f}_k)^{n-1}) = \mathcal{O}(\tilde{f}_k^{\,2n-1}).$$

As in Ref. [1], with a cost of a single additional function-evaluation, the convergence of the original n^{th}-order method has been raised to $2n-1$.

To complete contact with Ref. [1], we also calculate

$$(\Delta_k - \tilde{\delta}_k)\tilde{f}_k^{\,\prime} = (\tilde{g}_k - g^*)\tilde{f}_k - (1+\tilde{g}_k) f^* \sim -\tilde{f}_k \tilde{g}^{(1)} f^* - f^* \sim f^* \sim \mathcal{O}(\tilde{\delta}_k^n).$$

IV. INCREASING THE CONVERGENCE-RATE FURTHER WITHOUT ADDITIONAL DERIVATIVE EVALUATIONS

In this section we consider additional function evaluations (but no additional derivative evaluations) in going from f_k to f_{k+1} with a fixed-point iterative method. Our primary interest is to accelerate schemes for large systems (which typically call for $n=2$ to avoid the expense of derivatives beyond the first), but we begin with general n and with a scalar f for simplicity (the generalization to $M>1$ is straightforward).

We begin with some new notation. The subscript k will be dropped; all quantities should be understood to be at iteration k unless explicitly noted otherwise. Similarly the tilde ($\tilde{}$), denoting a modified method's iterates (as opposed to the original's, i.e., with Taylor series (5)) will be dropped. We identify step quantities with a bar ($\bar{}$) and the step index with a capitalized Roman-numeral superscript; and we define s to be the number of steps. We map our new variables onto the accelerated fixed-point method of Sec. III:

$$\begin{aligned}
\bar{x}^{\mathrm{S_0}} &\leftrightarrow \tilde{x}_k \\
\bar{f}^{\mathrm{S_0}} &\leftrightarrow \tilde{f}_k \\
\bar{A}^{\mathrm{S_0}} &\leftrightarrow \tilde{f}_k \\
\bar{g}^{\mathrm{S_0}} &\leftrightarrow \tilde{g}_k = g(\tilde{f}_k, \tilde{f}_k^{\,\prime}, \tilde{f}_k^{\,\prime\prime} \ldots) \\
\bar{\delta}^{\mathrm{S_0}} &\leftrightarrow \tilde{\delta}_k = -(\tilde{f}_k/\tilde{f}_k^{\,\prime}) \times (1+\tilde{g}_k) \\
\\
\bar{x}^{\mathrm{S_I}} &\leftrightarrow x^* = \tilde{x}_k + \tilde{\delta}_k \\
\bar{f}^{\mathrm{S_I}} &\leftrightarrow f^* = f(x^*) \\
\bar{A}^{\mathrm{S_I}} &\leftrightarrow \tilde{f}_k + f^* \\
\bar{g}^{\mathrm{S_I}} &\leftrightarrow g^* = g(\tilde{f}_k + f^*, \tilde{f}_k^{\,\prime}, \tilde{f}_k^{\,\prime\prime} \ldots) \\
\bar{\delta}^{\mathrm{S_I}} &\leftrightarrow \Delta_k = -((\tilde{f}_k + f^*)/\tilde{f}_k^{\,\prime}) \times (1+g^*) \\
\\
\bar{x}^{\mathrm{S_{II}}} &\leftrightarrow \tilde{x}_{k+1} = \tilde{x}_k + \Delta_k \\
\bar{f}^{\mathrm{S_{II}}} &\leftrightarrow \tilde{f}_{k+1} = f(\tilde{x}_{k+1}),
\end{aligned}$$

where we have introduced the new variable $\bar{A}^{\mathrm{S_i}} \equiv \sum_{j=0}^{i} \bar{f}^{\mathrm{S_j}}$.

We write the new scheme (omitting updates of quantities which simply follow their definitions, such as $\bar{f}^{\mathrm{S_{i+1}}} =$

$f(\bar{x}^{\mathrm{S}_{\mathrm{i+1}}})$):

$$\begin{aligned}
\bar{x}^{\mathrm{S}_0} &= \bar{x} \\
\bar{g}^{\mathrm{S}_0} &= g(\bar{A}^{\mathrm{S}_0}, f', f'', \dots) \\
\bar{\delta}^{\mathrm{S}_0} &= -(\bar{A}^{\mathrm{S}_0}/f') \times (1 + \bar{g}^{\mathrm{S}_0}) \\
\\
\bar{x}^{\mathrm{S}_\mathrm{i}} &= \bar{x} + \bar{\delta}^{\mathrm{S}_{\mathrm{i\text{-}1}}} \\
\bar{g}^{\mathrm{S}_\mathrm{i}} &= g(\bar{A}^{\mathrm{S}_\mathrm{i}}, f', f'', \dots) \\
\bar{\delta}^{\mathrm{S}_\mathrm{i}} &= -(\bar{A}^{\mathrm{S}_\mathrm{i}}/f') \times (1 + \bar{g}^{\mathrm{S}_\mathrm{i}}) \\
\\
\bar{x}^{\mathrm{S}_s} &= \bar{x} + \bar{\delta}^{\mathrm{S}_{s\text{-}1}} \\
x_{k+1} &= \bar{x}^{\mathrm{S}_s},
\end{aligned} \tag{7}$$

giving for the Taylor expansion of f_{k+1}:

$$f_{k+1} = \sum_{i=0}^{\infty} f^{(i)} \frac{(\bar{\delta}^{\mathrm{S}_{s\text{-}1}})^i}{i!} = f - (1 + \bar{g}^{\mathrm{S}_{s\text{-}1}})\bar{A}^{\mathrm{S}_{s\text{-}1}} + \sum_{i=2}^{\infty} \frac{f^{(i)}}{i!} \left(\frac{-\bar{A}^{\mathrm{S}_{s\text{-}1}}}{f'} \right)^i \left(1 + \bar{g}^{\mathrm{S}_{s\text{-}1}}\right)^i = f - \bar{A}^{\mathrm{S}_{s\text{-}1}} + \bar{h}^{\mathrm{S}_{s\text{-}1}}. \tag{8}$$

To proceed, we develop an iterative relation for the steps from the intermediate Taylor series, beginning with:

$$\bar{f}^{\mathrm{S}_{\mathrm{i+1}}} = f - \bar{A}^{\mathrm{S}_\mathrm{i}} + \bar{h}^{\mathrm{S}_\mathrm{i}}.$$

Subtracting the series for $\bar{f}^{\mathrm{S}_\mathrm{i}}$ gives:

$$\bar{f}^{\mathrm{S}_{\mathrm{i+1}}} - \bar{f}^{\mathrm{S}_\mathrm{i}} = -\bar{A}^{\mathrm{S}_\mathrm{i}} + \bar{A}^{\mathrm{S}_{\mathrm{i\text{-}1}}} + \bar{h}^{\mathrm{S}_\mathrm{i}} - \bar{h}^{\mathrm{S}_{\mathrm{i\text{-}1}}},$$

so that

$$\bar{f}^{\mathrm{S}_{\mathrm{i+1}}} = \bar{h}^{\mathrm{S}_\mathrm{i}} - \bar{h}^{\mathrm{S}_{\mathrm{i\text{-}1}}} \sim (\bar{A}^{\mathrm{S}_\mathrm{i}} - \bar{A}^{\mathrm{S}_{\mathrm{i\text{-}1}}})\, \mathcal{O}(f^{n-1}) \sim \bar{f}^{\mathrm{S}_\mathrm{i}}\, \mathcal{O}(f^{n-1}) \sim \mathcal{O}(f^{(i+1)(n-1)+1}),$$

where we have used Eq. (A7). Using this result in Eq. (8), we obtain the convergence rate:

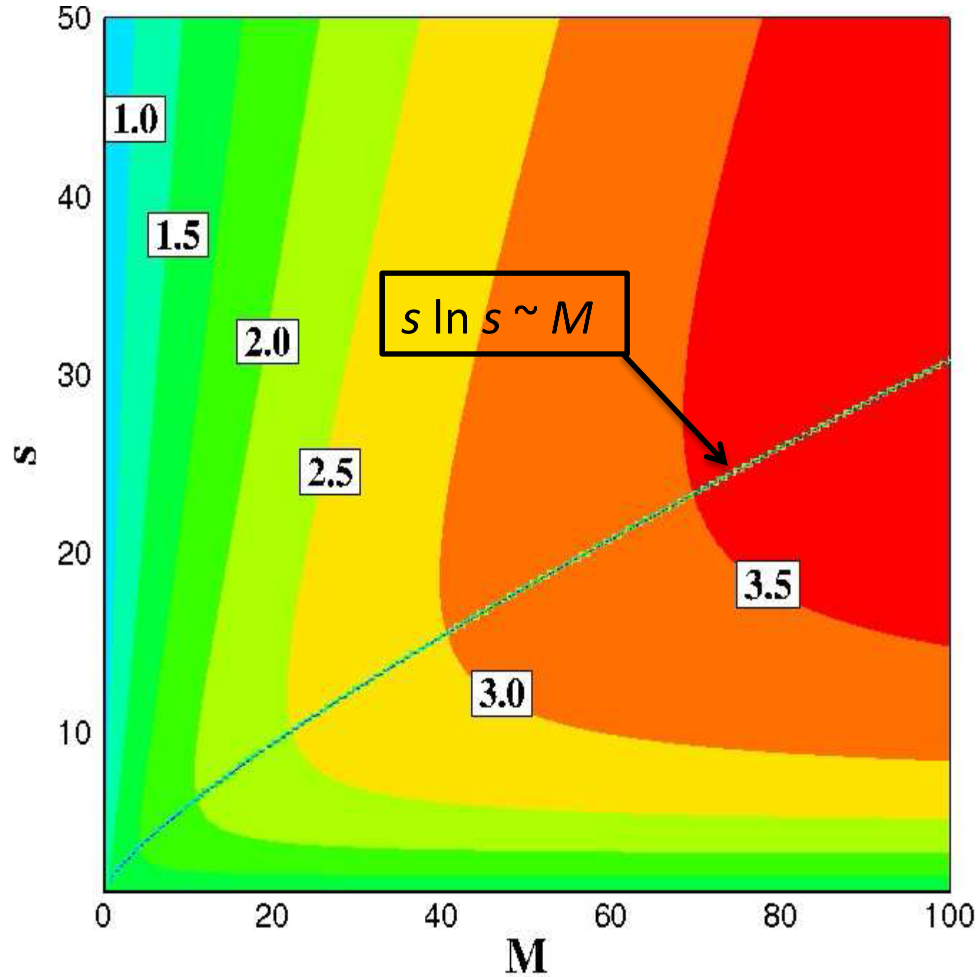
$$f_{k+1} \sim \mathcal{O}(f^{s(n-1)+1}), \tag{9}$$

having evaluated one derivative and s f-functions.

The error-reduction per function-evaluation of this scheme, assuming f' costs about the same as f to evaluate, is $\mathcal{E}_f = (s(n-1)+1)^{1/(n+s-1)}$. Evaluations of $\mathcal{E}_f$ at small s and n reveal a single maximum, $\mathcal{E}_f = 1.495$ at $s = 2$, $n = 3$, i.e., at the first case analyzed in [1]—Halley's scheme accelerated to 5$^{\mathrm{th}}$-order. For comparison, the original Halley's scheme, $s = 1$, $n = 3$, has $\mathcal{E}_f = 1.442$.

Accelerated Newton scheme shows continuous improvement with *s*

- Plotted are contours of constant $(\mathcal{E}_f(s,M)-1)/(\mathcal{E}_f(1,M)-1)$: the relative efficiency of the *s*-step scheme cf. Newton
  - Improvement over Newton is above a factor of 3 for *M* above 40.
  - $d\mathcal{E}_f/ds = 0 \Rightarrow s \ln s \sim M$: a good fit to the optimum *s* at given system-size.
  - Improvement continues as $M \longrightarrow \infty$.



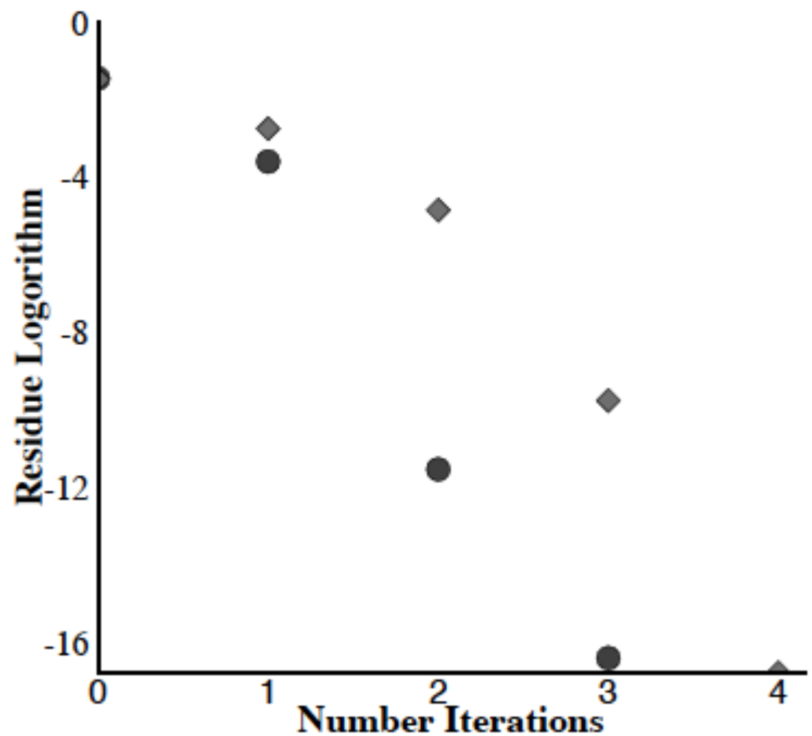
Example: tri-diagonal system

$$
\begin{aligned}
x_1 + \frac{1}{2}\sin(x_2) &= 1.0; \\
\frac{1}{2}\sin(x_1) \;+\; x_2 \;+\; \frac{1}{2}\sin(x_3) &= 1.0; \\
\ldots \quad \ldots &= \ldots \\
\frac{1}{2}\sin(x_{i-1}) \;+\; x_i \;+\; \frac{1}{2}\sin(x_{i+1}) &= 1.0; \\
\ldots \quad \ldots &= \ldots \\
\frac{1}{2}\sin(x_{M\text{-}1}) \;+\; x_M &= 1.0.
\end{aligned}
$$

- We solve this system with n=2, s=3 and M=32.
  - 8-byte arithmetic
  - Jacobian is inverted by back-substitution

tri-diagonal system, cont.

- Initial guess: x_i=1/2
- Residuals follow predicted convergence rates---parabola (diamonds) and cubic (circles) for Newton and accelerated Newton respectively.
  - Final iterations have reached machine accuracy.



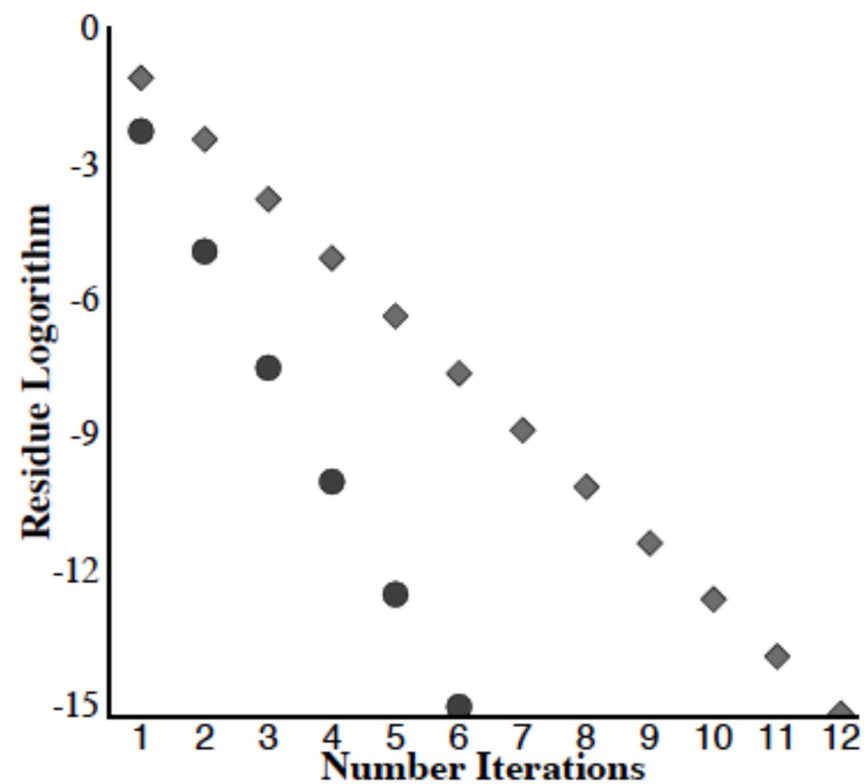
Example: ill-conditioned Jacobian

$$Mx_i + \sum_{j=1}^{M} \frac{\sin(x_i + x_j)}{i + j - 1} = \frac{M}{i} + \sum_{j=1}^{M} \frac{\sin(\frac{1}{i} + \frac{1}{j})}{i + j - 1}$$

- Solution is $x_i = 1/i$.
- We solve with n=2, s=3 and M=32.
  - 8-byte arithmetic
  - We use Cholesky decomposition of the Jacobian. Large condition-number ⇒ inverse is inaccurate

ill-conditioned Jacobian, cont.

- Initial guess: $x_i = x_{si} * (1+.5\ r_i)$
  - r_i = random number in [-1,1]
- Both the Newton and accelerated Newton schemes achieve only linear convergence due to finite accuracy of the Jacobian matrix decomposition.
- Accelerated scheme nevertheless still provides a significant advantage.



Summary/Discussion

- We have extended [Yao, JCP 2014 (s=2)] to an arbitrary number of steps s per iteration and estimated the error-reduction per function-evaluation, $\mathcal{E}_f$, and computational cost, C_{CPU}, of the methods.
  - No extra derivative-evaluations are needed.
  - The OoC is accelerated from the original order n to order $s(n\text{-}1)+1$.
  - M = 1: $\mathcal{E}_f = (s(n-1)+1)^{1/(n+s-1)}$.
    - At small n, $\mathcal{E}_f$ has a single maximum – at small s. For n=3, s_{max}=2 (which turns out to be the page 5 case). $\mathcal{E}_f$ goes from = 1.442 to 1.495 : small improvement.
  - $M > 1$, n=2: $\mathcal{E}_f = (s+1)^{1/(M(s+M))}$:
    - For $s \ll M$, $\mathcal{E}_f$ increases with s. Optimum s occurs at $s \ln s \sim M$: Leads to significant improvement in C_{CPU} for large M.
- Even when the Jacobian of an originally 2nd-order method is ill-conditioned and the theoretical OoC is not achieved, the new scheme can reduce the original C_{CPU} by a factor of two.